(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 526 424 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
10.12.2008 Bulletin 2008/50

(51) Int Cl.:
G06F 21/00 *(2006.01)* ***G06F 21/04*** *(2006.01)*

(21) Application number: **04020876.1**

(22) Date of filing: **02.09.2004**

(54) Providing a graphical user interface in a system with a high-assurance execution environment

Bereitstellung einer grafischen Benutzerschnittstelle in einem System mit gesicherter Ausführungsumgebung

Fourniture d'une interface utilisateur graphique dans un système avec un environnement d'exécution sécurisé

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PL PT RO SE SI SK TR

(30) Priority: **23.10.2003 US 691759**

(43) Date of publication of application:
27.04.2005 Bulletin 2005/17

(73) Proprietor: **MICROSOFT CORPORATION**
Redmond, Washington 98052 (US)

(72) Inventors:
- **Willman, Bryan, M.**
  Redmond
  Washington 98052 (US)
- **Chew, Christine, M.**
  Redmond
  Washington 98052 (US)
- **Avraham, Idan**
  Redmond
  Washington 98052 (US)
- **Roberts, Paul, C.**
  Redmond
  Washington 98052 (US)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser
Anwaltssozietät
Leopoldstrasse 4
80802 München (DE)

(56) References cited:
US-A- 5 590 266

- **EPSTEIN J ET AL: "User interface for a high assurance, windowing system" COMPUTER SECURITY APPLICATIONS CONFERENCE, 1993. PROCEEDINGS., NINTH ANNUAL ORLANDO, FL, USA 6-10 DEC. 1993, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, 6 December 1993 (1993-12-06), pages 256-264, XP010096756 ISBN: 0-8186-4330-7**
- **ZISHUANG YE, SEAN SMITH: "Trusted Paths for Browsers: An Open-Source Solution to Web Spoofing" TECHNICAL REPORT TR2002-418, [Online] 4 February 2002 (2002-02-04), XP002329977 DEPATMENT OF COMPUTER SCIENCE DARTMOUTH COLLEGE Retrieved from the Internet: URL:http: //www.cgisecurity.com/lib/tr418.p df> [retrieved on 2005-05-30]**
- **"AT WINHEC, MICROSOFT DISCUSSES DETAILS OF NEXT-GENERATION SECURE COMPUTING BASE"[Online] 7 May 2003 (2003-05-07), XP002302516 Retrieved from the Internet: URL: www.microsoft.com> [retrieved on 2004-10-26]**

EP 1 526 424 B1

Description

FIELD OF THE INVENTION

[0001] The present invention relates generally to the field of computer security. More particularly, the invention relates to the use of plural execution environments (e.g., operating systems) on a single computing device with a graphical user interface which allows graphical user interface elements to be owned by processes in each of the plural execution environments.

BACKGROUND OF THE INVENTION

[0002] In modem computing, many tasks which can be performed on a computer require some level of security. In order to provide a level of security, there are several options. One is to perform all secure applications on a computer which is completely separate from any possibly insecure elements, or to use a virtual machine monitor (VMM) to allow complete separation between two execution environments (e.g. operating systems) running on a single computer system. However, this may be impractical. There may be a need, for cost or convenience reasons, for a secure execution environment to share resources with applications with unassured security, and those applications and those resources may be vulnerable to an attacker. Additionally, where a VMM is used, since a VMM requires full virtualization of the machine and all of its devices (thereby requiring that the VMM provide its own device driver for every possible device), a VMM is not well suited to an open architecture machine in which an almost limitless variety of devices can be added to the machine.

[0003] One way to provide the ability to share resources among two execution environments is to provide a computer system in which there is one "main" operating system that controls most processes and devices on a machine, and where a second operating system also exists. This second operating system is a small, limited-purpose operating system alongside the main operating system which performs certain limited tasks. One way to make an operating system "small" or "limited-purpose" is to allow the small operating system to borrow certain infrastructure (e.g., the scheduling facility, the memory manager, the device drivers, etc.) from the "main" operating system. Since a VMM effectively isolates one operating system from another, this sharing of infrastructure is not practical using a VMM.

[0004] Certain other techniques allow operating systems to exist side-by-side on the same machine without the use of a VMM. One such technique is to have one operating system act as a "host" for the other operating system. (The operating system that the "host" is hosting is sometimes called a "guest.") In this case, the host operating system provides the guest with resources such as memory and processor time. Another such technique is the use of an "exokernel." An exokernel manages certain devices (e.g., the processor and the memory), and also manages certain types of interaction between the operating systems, although an exokernel - unlike a VMM - does not virtualize the entire machine. Even when an exokernel is used, it may be the case that one operating system (e.g., the "main" operating system) provides much of the infrastructure for the other, in which case the main operating system can still be referred to as the "host," and the smaller operating system as the "guest." Both the hosting model and the exokernel model allow useful types of interaction between operating systems that support sharing of infrastructure.

[0005] Thus, these techniques can be used to provide a computer system with at least two execution environments. One of these may be a "high-assurance" operating system, referred to herein as a "nexus." A high-assurance operating system is one that provides a certain level of assurance as to its behavior. For example, a nexus might be employed to work with secret information (e.g., cryptographic keys, etc.) that should not be divulged, by providing a curtained memory that is guaranteed not to leak information to the world outside of the nexus, and by permitting only certain certified applications to execute under the nexus and to access the curtained memory.

[0006] In a computer system with two execution environments, one of which is a nexus, it may be desirable for the nexus to be the guest operating system, and a second operating system, not subject to the same level of assurance as to behavior, to be the host operating system. This allows the nexus to be as small as possible. A small nexus allows a higher level of confidence in the assurance provided by the nexus. Therefore operating system functions are run by the host operating system.

[0007] One such operating system which may be run by the host operating system is a windowing system. When using a windowing system, a user's display will be populated with windows, areas on the screen which display information from an application. An application may have one or more windows.

[0008] When the windowing system is run by the host operating system, rather than by the nexus, it is vulnerable to attack. One such possible attack is known as a spoof. A spoof is an attack in which the user is led to believe that some hardware, system software, application or agent software, or a given window, is a trustworthy entity, even though it is not. The attacker is spoofing a trustworthy entity. This can be used to steal user credentials, or to capture other data of a sensitive nature entered by a user who thinks that the user is using a highly assured entity.

[0009] For example, in a system in which the nexus runs a banking program with a log-in screen, an attacker may write a program which runs on the host operating system, and displays a window which looks exactly like the log-in screen of the banking program. When the user is fooled by this spoof window, the user will enter information into the spoof window. This information is cap-

tured by the attacker and may then be used by the attacker without the knowledge of the user.

[0010] The windowing system is also vulnerable to an attack known as a snooker. In a snooker attack, the attacker changes the user display to make it appear to a user that the system is secure, when it is not. For example, a computer system may include the ability for a user to lock the system, or to allow the computer to sleep or hibernate. A snooker attack, in this case, would simulate the screen displayed when the system is locked, asleep, or hibernating. When the user turns their attention away, thinking that the system is inactive and secure, the attacker makes unauthorized use of the system.

[0011] Generally, whatever pattern of pixels a legitimate nexus-side program or functioning system can produce on the monitor, an attacking program on the host-side can imitate. However, in order to maintain the high assurance nature of the nexus, a user must be able to distinguish a legitimate nexus-side user interface graphic element from a fake one.

[0012] In their article "User Interface for a High Assurance Windowing System" (Computer Security Applications Conference, 1993. Proceedings., Ninth Annual), Epstein and Pascale describe graphical user interfaces which can accommodate multiple levels of security. The authors suggest having each of the windows sharing the display include indicators of the corresponding level of security, such as window labels (e.g.,"top secret") or window borders in reserved colors. In conjunction with tiling restrictions preventing total or partial overlap of windows, such indicators can be useful for preventing spoofing of a window's security level. It is to be inferred, however, that said indicators are to be chosen in a consistent, easy-to-guess way, which would give malicious windows the opportunity to spoof secure ones by mimicking their security indicators.

[0013] In view of the foregoing there is a need for a system that overcomes the drawbacks of the prior art.

SUMMARY OF THE INVENTION

[0014] In one embodiment, data displayed on a display for a system including a secured execution environment (nexus) and a second execution environment (host) is secured using a combination of several techniques. Graphical user interface elements, such as windows, which are associated with a process running on the nexus are displayed without overlap from other graphical user interface elements.

[0015] Additionally, a nexus-user secret is maintained, which is displayed in the graphical user interface element. This display can occur continuously or, among other alternatives, on request. Additionally, window decorations may be coordinated, in color or in graphics information being displayed, in order to link together in a user's mind the secured windows and thus more clearly highlight impostor windows. These decorations can be changed at certain intervals, upon request, or when a system event triggers the change.

[0016] Where a shadow window is maintained for corresponding nexus windows, private title information may be used for the nexus window while a public title is used in the shadow window. This allows the nexus process which sets the titles to choose what information will be given to the possibly insecure shadow window.

[0017] Other features of the invention are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The foregoing summary, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings exemplary constructions of the invention; however, the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

FIG. 1 is a block diagram of an exemplary computing environment in which aspects of the invention may be implemented;
FIG. 2 is a block diagram of two exemplary execution environments that maintain some interaction with each other and some separation from each other;
FIG. 3 (a) is a block diagram of a display;
FIG. 3(b) is a block diagram of a display according to one embodiment of the invention;
FIG. 4 is a flow diagram of a method for maintaining the security of data displayed on a display;
FIG. 5 is a flow diagram of a method for maintaining the security of data displayed on a display;
FIG. 6 is a block diagram of a display according to one embodiment of the invention; and
FIG. 7 is a flow diagram of a method for maintaining the security of data displayed on a display;
FIG. 8 is a flow diagram of a method for maintaining the security of data displayed on a display.

DETAILED DESCRIPTION OF THE INVENTION

Overview

[0019] When two execution environments, such as operating systems, run side-by-side on a single machine, where one of the environments is a high-assurance execution environment which is held to certain standards of security, it may be important for a user to discern which graphical user interface elements being displayed are associated with processes running on the high-assurance operating system. As discussed above, an attacker displaying a graphical UI element via the host side (non high-assurance) execution environment may attempt to convince a user that a graphical user interface element that is a UI element arising from a high assurance process. In order to prevent such attacks, the present inven-

tion provides techniques that allow a user to discern which graphical user interface elements originate in the high-assurance execution environment.

Exemplary Computing Arrangement

[0020] FIG. 1 shows an exemplary computing environment in which aspects of the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0021] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, embedded systems, distributed computing environments that include any of the above systems or devices, and the like.

[0022] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network or other data transmission medium. In a distributed computing environment, program modules and other data may be located in both local and remote computer storage media including memory storage devices.

[0023] With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The processing unit 120 may represent multiple logical processing units such as those supported on a multi-threaded processor. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus). The system bus 121 may also be implemented as a point-to-point connection, switching fabric, or the like, among the communicating devices.

[0024] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0025] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0026] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 140 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155

that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0027] The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 190.

[0028] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0029] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Plural Computing Environments on a Single Machine

[0030] As previously described, it is known in the art that two operating systems can execute side-by-side on a single computing device. One problem that the present invention can be used to address is how to provided some level of separation between two operating system, while still providing for some level of interaction between the two operating systems.

[0031] FIG. 2 shows a system in which two operating systems 134(1) and 134(2) execute on a single computer 110. Some type of logical separation 202 exists between operating systems 134(1) and 134(2), such that a certain amount of interaction 204 is permitted between operating systems 134(1) and 134(2), while still allowing at least one of the operating systems to be protected against events that originate in the other operating system. In the example of FIG. 2, operating system 134(1) is a host operating system, and operating system 134(2) is a guest operating system, such as a "nexus" as described above. As previously noted, when operating system 134(2) is a nexus, it is desirable to construct separation 202 such that operating system 134(2) can interact with operating system 134(1) in order to borrow operating system 134(1)'s infrastructure, while still allowing operating system 134(2) to protect itself from actions (either malicious or innocent) that arise at operating system 134(1) and might cause operating system 134(2) to behave in a manner contrary to its behavioral specifications. (It will be understood, however, that the invention is not limited to the case where operating system 134(2) is a nexus.)

[0032] The separation 202 between operating systems 134(1) and 134(2) may, optionally, be enforced with the aid of a security monitor. A security monitor is a component external to both operating systems 134(1) and 134(2), which provides some security services that may be used to protect operating system 134(2) from operating system 134(1). For example, a security monitor may control access to certain hardware, may manage the use of memory (to give operating system 134(2) exclusive use

of some portions of memory), or may facilitate the communication of data from operating system 134(1) to operating system 134(2) in a secure way. It should be noted that the use of a security monitor represents one model of how operating system 134(2) can be protected from operating system 134(1), although the use of a security monitor is not required. As another example, operating system 134(2) could include all of the functionality necessary to protect itself from operating system 134(1).

[0033] It should be noted that FIG. 2 shows operating system 134(1) as a "host" and operating system 134(2) as a "guest." In general, this characterization refers to the fact that, in these examples, operating system 134 (1) provides certain operating system infrastructure that is used by both operating systems 134(1) and 134(2) (e.g., device drivers, scheduling, etc.), and operating system 134(2) is a "guest" in the sense that it preferably lacks this infrastructure but rather uses the infrastructure of operating system 134(1). However, it should be noted that the parameters of what makes an operating system a "host" or a "guest" are flexible. Moreover, it should be noted that traditional concepts of "host" and "guest" operating systems presume that the host needs to protect itself from actions of the guest. In the example of FIGS. 2, however, guest operating system 134(2) is presumed to be a high-assurance operating system that needs to protect itself from host operating system 134(1). In the examples that follow, we shall generally refer to operating system 134(1) as the "host" and operating system 134 (2) as the "guest" or "nexus" for the purpose of distinguishing between them. It should be appreciated that the techniques described herein can be applied to the interaction of any two or more operating systems running on the same machine (or even on the same set of connected machines).

Providing A Graphical User Interface With Plural Computing Environments On A Single Machine

[0034] When a user interacts with programs on a computer system containing a high-assurance operating system, the user does so by means of a user input device, such as mouse 161 or keyboard 162 (from Figure 1). As discussed above, allowing the windowing system running on host operating system 134(1) control the destination of the stream of input events may allow an attack using a compromised host operating system or application. A window or other graphical user interface element may be displayed by a host-side process to the user, which may attempt to imitate a legitimate nexus-side process, running on nexus operating system 134(2). Therefore, according to one embodiment of the invention, several techniques for protecting the integrity and identifiableness of legitimate host-side windows and other graphical user interface elements are implemented. In various embodiments of the invention, any one or all of these techniques are implemented together.

Obscuration and Show-Through

[0035] Where both host-side and nexus-side processes can display graphics, such as windows and other user interface elements, on a display, an attack can be accomplished by obscuring all or part of a legitimate nexus-side process graphic with a host-side process graphic. Figure 3(a) is a block diagram of a display. In Figure 3 (a), on display 300, nexus graphics 310 are graphics owned by nexus processes, running on the nexus operating system 134(2). Host graphics 320 are graphics owned by host processes, running on the host operating system 134(1). As shown in Figure 3 (a), where host graphics 320 are allowed to be displayed on top of nexus graphics 310, the nexus graphics may be obscured by the host graphics 320. In this case, an attack may be mounted by obscuring some or all of nexus graphic 310 with a host graphic 320. Additionally, if the nexus graphic 310 overlays a host graphic 320 but is partially transparent, the host graphic 320 may be used to change the appearance of the nexus graphic 310 in a confusing way.

[0036] In one embodiment, such attacks are prevented by preventing the all nexus graphics 310 from being obscured, and prohibiting any show-through in nexus graphics 310. Thus, for the situation shown in Figure 3 (a), the nexus graphics 310 would not be allowed to be obscured. Figure 3(b) is an illustration of the prohibition on obscuration of nexus graphic 310. In Figure 3(b), each of the two nexus graphics 310 are fully shown, and no overlap by a host graphic 320 of a nexus graphic 310 is permitted. In addition, in one embodiment, no transparency (either full transparency or partial transparency) is permitted in a nexus graphic 310.

[0037] In one embodiment, in order to prevent similar attacks from one nexus-side process on another, nexus graphics 310 may not overlap. In a windowing system, one embodiment allows one exception to this - the mouse cursor, which may be drawn by a nexus-side process when being displayed over a nexus graphic 310, may be allowed to overlap a nexus-side graphic 310.

[0038] In another embodiment, if one nexus-side process owns two or more user interface graphic elements, for example two windows or a window and a dialog box, no security issue is implicated in allowing these to overlap. Thus, one commonly-owned user interface graphic element is allowed to overlap another commonly-owned user interface graphic element owned by a nexus process. In still another embodiment, top-level graphic UI elements which are commonly-owned are not permitted to overlap, however, a top-level UI element may be overlapped by a child UI element of that top-level UI element. For example, in this embodiment, a dialog box which is a child UI element of a top-level window can overlap that window.

[0039] In one embodiment, in order to verify which user interface graphic elements are nexus-side user interface graphic elements, a hypothecated user interaction is provided which removes all non-nexus-side user interface

graphic elements from the screen. A hypothecated user interaction is a user interaction which, in the context of the computer system, will always result in a specific consequence. Thus, when the secure display hypothecated user interaction, for example, a keystroke combination, occurs, the entire display is cleared, with the exception of nexus-side user interface graphic elements.

[0040] Figure 4 is a flow diagram of this method. In step 400, a nexus graphical user interface element image, associated with a process running on the secured execution environment (nexus agent) is stored. In step 410, the nexus graphical user interface element image is displayed completely, unobscured by any host user interface elements. In other embodiments, no graphical user interface elements (host or user) other than those associated with the process can obscure the nexus graphical user interface element.

Secret Sharing

[0041] In one embodiment, in order to prevent the spoofing attack described above, a secret is displayed which is hidden from the host-side. No process on the host side can access the secret, and therefore if a window or other graphic user interface element can display the secret, it is a host-side graphic user interface element.

[0042] In one embodiment, a secret is communicated to the nexus by the user. This nexus-user secret may be communicated to the nexus at trust-initialization time, for example when passwords are set by the user. The graphic user interface element may either display the nexus-user secret at all times or may display the nexus-user secret when challenged.

[0043] A trusted window manager on the nexus side controls the display of graphic user interface elements by processes on the nexus side. This trusted window manager is also responsible for window decorations, such as borders and titles of windows. In one embodiment, the nexus-user secret is not generally shared with processes in the nexus. It is, however, displayed in the window decoration by the trusted window manager. In an alternate embodiment, the nexus-user secret is displayed when a user requests it from the window. This request may be made on the window in focus through a hypothecated user interaction. For example, a specific combination of keystrokes may be the hypothecated user interaction which causes the window in focus to display the nexus-user secret. Alternatively, a window may include a drop down menu or activation area which, when selected by the mouse or by keystrokes causes a valid nexus graphic 310 to display the nexus-user secret.

[0044] The nexus-user secret may be an image or a phrase. An image can be a useful nexus-user secret, because they are easily identified by the user and hard to describe. If the image selected by the user for use as the nexus-user secret is, for example, a photograph of the user's dog in front of the user's house, the photograph may be described by an attacker who views the image on the user's screen, however, even with that information an attacker would have difficulty recreating the image or finding a copy of it.

[0045] Figure 5 is a flow diagram of this method. In step 500, a nexus-user secret associated with the nexus is stored. In step 510, the nexus-user secret element image is displayed as part of a nexus graphical user interface element.

Continual Secret Permutation

[0046] As described, in one embodiment a trusted window manager is the intermediary through which all nexus-side processes display their user interface graphic elements, and the trusted window manager is solely responsible for the window decoration, such as borders, on user interface graphic elements. In one embodiment, the window decoration includes a continually-updated short term secret. This continually updated secret is not accessible to non-nexus windows, and can thus be used to identify a non-host window.

[0047] For example, where the border of each nexus user interface graphic element is a specific color, and that color changes once every 15 seconds, it will be obvious to a user when a given window border does not match other nexus user interface graphic elements that that window is not a nexus-side user interface graphic elements. Other possible continually updated secrets may include: small graphics, combinations of icons, glyphs, or characters displayed in the window decoration, or hexadecimal or numeric strings. Any piece of information which can be visually compared with discrepancies noted fairly easily by a user might be used.

[0048] In one embodiment, a nexus system user interface area is maintained on the display 300 whenever a nexus graphic 310 is displayed. This user interface area lists information regarding the nexus graphics 310 being displayed. For example, the user interface area may list the number of nexus graphics 310 being displayed, or the names of the nexus graphics 310 being displayed (e.g. window titles.) The shared secret in the decoration of the nexus graphics 310 is also displayed in the nexus system user interface area. Figure 6 is a block diagram of a display according to one embodiment of the invention. In Figure 6, a series consisting of three glyphs ("♨♦□") is displayed in window decoration 610 of each nexus graphic 310 on the display 300. A nexus system interface area 600 is also displayed, which includes the same set of three glyphs. This allows easy user confirmation that a window is being displayed by a nexus-side process.

[0049] While the changing of the secret may be time-based, such as a colored border that changes every 15 seconds, as described above, in other embodiments, the secret permutes when the user requests it, through a hypothecated user interaction, or when a system event such as the change of focus from one window to another occurs.

[0050] Figure 7 is a flow diagram of this method. In step 700, at least two nexus graphical data elements are accepted. In a windowing system, these, along with the window decorations, will constitute a window image. In step 710, two nexus graphical user interface elements (e.g. windows) are displayed - each comprising one of the graphical data elements in addition to decoration which is common to each window.

Window Titles - Public and Private

[0051] As described, a trusted window manager can be used as an intermediary through which all nexus-side processes display their user interface graphic elements. However, a host-side window manager can be used to manage the display of host-side processes' user interface graphic elements. In one embodiment, for each nexus-side user interface graphic element (such as nexus graphic 310) a corresponding shadow graphic user interface elements is maintained by the host-side window manager. This allows the host-side window manager to recognize that certain areas of the display 300 are being used by nexus graphics 310. When obscuration and show-through are prohibited, as described above, this may be useful information for the host-side window manager, so windows which should be visible are not placed in these areas. However, only limited information about nexus graphics 310 should be available in the host-side.

[0052] In one embodiment, therefore, the trusted window manager allows a nexus agent (a process running on the nexus) to set two titles for each window or other user interface graphic element. One title, the private title, is passed to the trusted window manager and used for window management functions and for display in nexus graphic 310. This private title is not shared by the trusted window manager with any host-side processes or with any nexus-side processes other than the nexus agent which owns the window (or other element.)

[0053] In addition to the private title, the nexus agent can also specify a public title. This public title may be shared with the host-side, including the host-side window manager. The public title may be used by the host-side window manager as the title for the shadow window. In this way, the host-side window manager can reference the window using the public title, which is selected in order to be understandable to the user without including information which may breach confidentiality. Thus, where the host-side window manager allows a user to select a window to focus on from a list of windows, the public title, associated with the shadow window, is listed as an option to select. When the shadow window is selected, the associated trusted window will be activated, and the nexus agent associated with the associated trusted window will become the focus of user input.

[0054] Figure 8 is a flow diagram of this method. In step 800, public title information and private title information is stored for a nexus graphical user interface element associated with a nexus agent. In step 810, the private title information is used for window management functions of the trusted window manager. In step 820, the public title information is provided to the host for use on the host-side.

Conclusion

[0055] It is noted that the foregoing examples have been provided merely for the purpose of explanation and are in no way to be construed as limiting of the present invention. While the invention has been described with reference to various embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of limitations. Further, although the invention has been described herein with reference to particular means, materials and embodiments, the invention is not intended to be limited to the particulars disclosed herein; rather, the invention extends to all functionally equivalent structures, methods and uses, such as are within the scope of the appended claims.

Claims

1. A method for maintaining the security of data displayed on a display for a system comprising a secured execution environment and a second execution environment, comprising:

   storing an image of at least one first graphical user interface element, associated with a first process running on said secured execution environment; and
   displaying said first graphical user interface element, including a first secret associated with said secured execution environment, on said display completely on a display, such that no part of said first graphical user interface element is obscured by a second graphical user interface element associated with said second execution environment on said display, said first secret being communicated to said secured execution environment by a user and not being accessible by a second process on said second execution environment.

2. The method of claim 1, where said step of displaying said first graphical user interface element comprises:

   ensuring that said first graphical user interface element contains no areas of transparency.

3. The method of claim 1, where said step of displaying said first graphical user interface element on a display comprises displaying said first graphical user interface element such that no part of said first graphical user interface element is obscured by said sec-

ond graphical user interface element, which is associated with a third process running on said secured execution environment.

4. The method of claim 1, further comprising:

displaying only said first graphical user interface element on said display upon receipt of a user secure display indication.

5. The method of claim 1, further comprising:

accepting a secret-display indication; from a user; and
displaying said first secret in response to acceptance of said secret-display indication.

6. The method of claim 1, further comprising:

accepting at least two graphical data elements, each associated with a process running on said secured execution environment, for display on said display; and
displaying at least two graphical user interface elements, each of said graphical user interface elements comprising one of said graphical data elements and a common graphical user interface decoration.

7. The method of claim 6, where said common graphical user interface decoration comprises a colored border.

8. The method of claim 6, where said common graphical user interface decoration comprises one or more randomly selected images.

9. The method of claim 6, further comprising:

changing said common graphical user interface decoration when a set time period elapses.

10. The method of claim 6, further comprising:

changing said common graphical user interface decoration when a user decoration change indication is received.

11. The method of claim 1, wherein said first graphical user element comprises a window, wherein said first graphical user element is associated with public title information and private title information, wherein said private title information is used to identify said window in said secured execution environment, wherein said private title information is not known to said second execution environment, and wherein the method further comprises:

providing said public title information for use by said second execution environment to identify said window.

12. The method of claim 11, where said second execution environment includes a host window manager for managing graphical user interface elements on said display, where said host window manager creates a shadow graphical user interface element for said first graphical user interface element, and where said public title is used by said host window manager.

13. A computer-readable medium containing computer executable instructions to maintain the security of data displayed on a display for a system comprising a secured execution environment and a second execution environment, the computer-executable instructions to perform acts comprising:

storing an image of at least one first graphical user interface element associated with a first process running on said secured execution environment; and
displaying said first graphical user interface element, including a first secret associated with said secured execution environment, on said display completely on a display, such that no part of said first graphical user interface element is obscured by a second graphical user interface element associated with said second execution environment on said display, said first secret being communicated to said secured execution environment by a user and not being accessible by a second process on said second execution environment.

14. The computer-readable medium of claim 13, where said act of displaying said first graphical user interface element comprises:

ensuring that said first graphical user interface element contains no areas of transparency.

15. The computer-readable medium of claim 13, where said act of displaying said first graphical user interface element on a display comprises displaying said first graphical user interface element such that no part of said first graphical user interface element is obscured by said second graphical user interface element, which is associated with a third process running on said secured execution environment.

16. The computer-readable medium of claim 13, wherein the computer-executable instructions are adapted to perform acts further comprising:

displaying only said first graphical user interface element on said display upon receipt of a user

secure display indication.

17. The computer-readable medium of claim 13, further comprising:

accepting a secret-display indication from a user; and
displaying said first secret in response to acceptance of said secret-display.

18. The computer-readable medium of claim 13, wherein the computer-executable instructions are adapted to perform acts further comprising:

accepting at least two graphical data elements, each associated with a process running on said secured execution environment, for display on said display; and
displaying at least two graphical user interface elements, each of said graphical user interface elements comprising one of said graphical data elements and a common graphical user interface decoration.

19. The computer-readable medium of claim 18, where said common graphical user interface decoration comprises a colored border.

20. The computer-readable medium of claim 18, where said common graphical user interface decoration comprises one or more randomly selected images.

21. The computer-readable medium of claim 18, wherein the computer-executable instructions are adapted to perform acts further comprising:

changing said common graphical user interface decoration when a set time period elapses.

22. The computer-readable medium of claim 18, wherein the computer-executable instructions are adapted to perform acts further comprising:

changing said common graphical user interface decoration when a user decoration change indication is received.

23. The computer-readable medium of claim 13, wherein said first graphical user element comprises a window, wherein said first graphical user element is associated with public title information and private title information, wherein said private title information is used to identify said window in said secured execution environment, wherein said private title information is not known to said second execution environment, and wherein the acts performable by the computer-executable instructions further comprise:

providing said public title information for use by said second execution environment to identify said window.

24. The computer-readable medium of claim 23, where said second execution environment includes a host window manager for managing graphical user interface elements on said display, where said host window manager creates a shadow graphical user interface element for said first graphical user interface element, and where said public title is used by said host window manager.

Patentansprüche

1. Verfahren zum Aufrechterhalten der Sicherheit von Daten, die auf einer Anzeigeeinrichtung für ein System angezeigt werden, das eine gesicherte Ausführungsumgebung und eine zweite Ausführungsumgebung umfasst, wobei das Verfahren umfasst:

Speichern eines Abbildes wenigstens eines ersten grafischen Benutzerschnittstellenelementes, das mit einem ersten Prozess zusammenhängt, der in der gesicherten Ausführungsumgebung läuft; und
vollständiges Anzeigen des ersten grafischen Benutzerschnittstellenelementes, das ein erstes Geheimnis enthält, das mit der gesicherten Ausführungsumgebung zusammenhängt, auf der Anzeigeeinrichtung, so dass kein Teil des ersten grafischen Benutzerschnittstellenelementes durch ein zweites grafischen Benutzerschnittstellenelement, das mit der zweiten Ausführungsumgebung zusammenhängt, auf der Anzeigeeinrichtung verdeckt wird, wobei das erste Geheimnis durch einen Benutzer zu der gesicherten Ausführungsumgebung übertragen wird und für einen zweiten Prozess in der zweiten Ausführungsumgebung nicht zugänglich ist.

2. Verfahren nach Anspruch 1, wobei der Schritt des Anzeigens des ersten grafischen Benutzerschnittstellenelementes umfasst:

Gewährleisten, dass das erste grafische Benutzerschnittstellenelement keine Bereiche von Transparenz enthält.

3. Verfahren nach Anspruch 1, wobei der Schritt des Anzeigens des ersten grafischen Benutzerschnittstellenelementes auf einer Anzeigeeinrichtung umfasst, dass das erste grafische Benutzerschnittstellenelement so angezeigt wird, dass kein Teil des ersten grafischen Benutzerschnittstellenelementes durch das zweite grafische Benutzerschnittstellenelement verdeckt wird, das mit einem dritten Prozess

zusammenhängt, der in der gesicherten Ausführungsumgebung läuft.

4. Verfahren nach Anspruch 1, das des Weiteren umfasst:

Anzeigen nur des ersten grafischen Benutzerschnittstellenelementes auf der Anzeigeeinrichtung beim Empfang eines Hinweises eines Benutzers auf sichere Anzeige.

5. Verfahren nach Anspruch 1, das des Weiteren umfasst:

Annehmen eines Hinweises von einem Benutzer auf Geheimnisanzeige, und
Anzeigen des ersten Geheimnisses in Reaktion auf Annahme des Hinweises auf Geheimnisanzeige.

6. Verfahren nach Anspruch 1, das des Weiteren umfasst:

Annehmen von wenigstens zwei grafischen Datenelementen, die jeweils mit einem Prozess zusammenhängen, der in der gesicherten Ausführungsumgebung läuft, zur Anzeige auf der Anzeigeeinrichtung; und
Anzeigen von wenigstens zwei grafischen Benutzerschnittstellenelementen, wobei jedes der grafischen Benutzerschnittstellenelemente eines der grafischen Datenelemente und eine gemeinsame grafische Benutzerschnittstellen-Dekoration umfasst.

7. Verfahren nach Anspruch 6, wobei die gemeinsame grafische Benutzerschnittstellen-Dekoration einen gefärbten Rand umfasst.

8. Verfahren nach Anspruch 6, wobei die gemeinsame grafische Benutzerschnittstellen-Dekoration ein oder mehr willkürlich ausgewählte/s Bild/er umfasst.

9. Verfahren nach Anspruch 6, das des Weiteren umfasst:

Ändern der gemeinsamen grafischen Benutzerschnittstellen-Dekoration, wenn ein festgelegter Zeitraum abläuft.

10. Verfahren nach Anspruch 6, das des Weiteren umfasst:

Ändern der gemeinsamen grafischen Benutzerschnittstellen-Dekoration, wenn ein Hinweis eines Benutzer auf Änderung der Dekoration empfangen wird.

11. Verfahren nach Anspruch 1, wobei das erste grafische Benutzerelement ein Fenster umfasst, das erste grafische Benutzerelement mit öffentlichen Titelinformationen und privaten Titelinformationen zusammenhängt, die privaten Titelinformationen dazu dienen, das Fenster in der gesicherten Ausführungsumgebung zu identifizieren, die privaten Titelinformationen der zweiten Ausführungsumgebung nicht bekannt sind und wobei das Verfahren des Weiteren umfasst:

Bereitstellen der öffentlichen Titelinformationen zur Verwendung durch die zweite Ausführungsumgebung zum Identifizieren des Fensters.

12. Verfahren nach Anspruch 11, wobei die zweite Ausführungsumgebung einen Host-Fenster-Manager zum Verwalten grafischer Benutzerschnittstellenelemente auf der Anzeigeeinrichtung enthält, der Host-Fenster-Manager ein schattiertes grafisches Benutzerschnittstellenelement für das erste grafische Benutzerschnittstellenelement erzeugt und der öffentliche Titel von dem Host-Fenster-Manager verwendet wird.

13. Computerlesbares Medium, das durch Computer ausführbare Befehle zum Aufrechterhalten der Sicherheit von Daten enthält, die auf einer Anzeigeeinrichtung für ein System angezeigt werden, das eine gesicherte Ausführungsumgebung und eine zweite Ausführungsumgebung umfasst, wobei die durch Computer ausführbaren Befehle dazu dienen, Vorgänge durchzuführen, die umfassen:

Speichern eines Abbildes wenigstens eines ersten grafischen Benutzerschnittstellenelementes, das mit einem ersten Prozess zusammenhängt, der in der gesicherten Ausführungsumgebung läuft; und
vollständiges Anzeigen des ersten grafischen Benutzerschnittstellenelementes, das ein erstes Geheimnis enthält, das mit der gesicherten Ausführungsumgebung zusammenhängt, auf der Anzeigeeinrichtung, so dass kein Teil des ersten grafischen Benutzerschnittstellenelementes durch ein zweites grafisches Benutzerschnittstellenelement, das mit der zweiten Ausführungsumgebung zusammenhängt, auf der Anzeigeeinrichtung verdeckt wird, wobei das erste Geheimnis durch einen Benutzer zu der gesicherten Ausführungsumgebung übertragen wird und für einen zweiten Prozess in der zweiten Ausführungsumgebung nicht zugänglich ist.

14. Computerlesbares Medium nach Anspruch 13, wobei der Vorgang des Anzeigens des ersten grafischen Benutzerschnittstellenelementes umfasst:

Gewährleisten, dass das erste grafische Benutzerschnittstellenelement keine Bereiche von Transparenz enthält.

15. Computerlesbares Medium nach Anspruch 13, wobei der Vorgang des Anzeigens des ersten grafischen Benutzerschnittstellenelementes auf einer Anzeigeeinrichtung umfasst, dass das erste grafische Benutzerschnittstellenelement so angezeigt wird, dass kein Teil des ersten grafischen Benutzerschnittstellenelementes durch das zweite grafische Benutzerschnittstellenelement verdeckt wird, das mit einem dritten Prozess zusammenhängt, der in der gesicherten Ausführungsumgebung läuft.

16. Computerlesbares Medium nach Anspruch 13, wobei die durch Computer ausführbaren Befehle zum Durchführen von Vorgängen eingerichtet sind, die des Weiteren umfassen:

Anzeigen nur des ersten grafischen Benutzerschnittstellenelementes auf der Anzeigeeinrichtung beim Empfang eines Hinweises eines Benutzers auf sichere Anzeige.

17. Computerlesbares Medium nach Anspruch 13, das des Weiteren umfasst:

Annehmen eines Hinweises von einem Benutzer auf Geheimnisanzeige; und
Anzeigen des ersten Geheimnisses in Reaktion auf Annahme der Geheimnisanzeige.

18. Computerlesbares Medium nach Anspruch 13, wobei die durch Computer ausführbaren Befehle zum Durchführen von Vorgängen eingerichtet sind, die des Weiteren umfassen:

Annehmen von wenigstens zwei grafischen Datenelementen, die jeweils mit einem Prozess zusammenhängen, der in der gesicherten Ausführungsumgebung läuft, zur Anzeige auf der Anzeigeeinrichtung; und
Anzeigen von wenigstens zwei grafischen Benutzerschnittstellenelementen, wobei jedes der grafischen Benutzerschnittstellenelemente eines der grafischen Datenelemente und eine gemeinsame grafische Benutzerschnittstellen-Dekoration umfasst.

19. Computerlesbares Medium nach Anspruch 18, wobei die gemeinsame grafische Benutzerschnittstellen-Dekoration einen farbigen Rand umfasst.

20. Computerlesbares Medium nach Anspruch 18, wobei die gemeinsame grafische Benutzerschnittstellen-Dekoration ein oder mehr willkürlich ausgewählte/s Bild/er umfasst.

21. Computerlesbares Medium nach Anspruch 18, wobei die durch Computer ausführbaren Befehle zum Durchführen von Vorgängen eingerichtet sind, die des Weiteren umfassen:

Ändern der gemeinsamen grafischen Benutzerschnittstellen-Dekoration, wenn ein festgelegter Zeitraum abläuft.

22. Computerlesbares Medium nach Anspruch 18, wobei die durch Computer ausführbaren Befehle zum Durchführen von Vorgängen eingerichtet sind, die des Weiteren umfassen:

Ändern der gemeinsamen grafischen Benutzerschnittstellen-Dekoration, wenn ein Hinweis eines Benutzers auf Änderung der Dekoration empfangen wird.

23. Computerlesbares Medium nach Anspruch 13, wobei das erste grafische Benutzerelement ein Fenster umfasst, das erste grafische Benutzerelement mit öffentlichen Titelinformationen und privaten Titelinformationen zusammenhängt, die privaten Titelinformationen dazu dienen, das Fenster in der gesicherten Ausführungsumgebung zu identifizieren, die privaten Titelinformationen der zweiten Ausführungsumgebung nicht bekannt sind, und wobei die durch die Computer ausführbaren Befehlen durchführbaren Vorgänge des Weiteren umfassen:

Bereitstellen der öffentlichen Titelinformationen zur Verwendung durch die zweite Ausführungsumgebung zum Identifizieren des Fensters.

24. Computerlesbares Medium nach Anspruch 23, wobei die zweite Ausführungsumgebung einen Host-Fenster-Manager zum Verwalten grafischer Benutzerschnittstellenelemente auf der Anzeigeeinrichtung enthält, der Host-Fenster-Manager ein schattiertes grafisches Benutzerschnittstellenelement für das erste grafische Benutzerschnittstellenelement erzeugt und der öffentliche Titel von dem Host-Fenster-Manager verwendet wird.

Revendications

1. Procédé pour maintenir la sécurité de données affichées sur un dispositif d'affichage pour un système comportant un environnement d'exécution sécurisé et un second environnement d'exécution, comprenant :

le stockage d'une image d'au moins un premier élément d'interface graphique utilisateur associé à un premier processus exécuté dans ledit environnement d'exécution sécurisé ; et

l'affichage dudit premier élément d'interface graphique utilisateur qui comprend un premier secret associé audit environnement d'exécution sécurisé, sur ledit dispositif d'affichage entièrement sur un écran, de façon qu'aucune partie dudit premier élément d'interface graphique utilisateur ne soit cachée par un second élément d'interface graphique utilisateur associé audit second environnement d'exécution sur ledit dispositif d'affichage, ledit premier secret étant communiqué audit environnement d'exécution sécurisé par un utilisateur et n'étant pas accessible par un second processus exécuté dans ledit second environnement d'exécution.

2. Procédé selon la revendication 1, dans lequel ladite étape d'affichage dudit premier élément d'interface graphique utilisateur comprend :

le fait de faire en sorte que ledit premier élément d'interface graphique utilisateur ne contienne aucune zone de transparence.

3. Procédé selon la revendication 1, dans lequel ladite étape d'affichage dudit premier élément d'interface graphique utilisateur sur un dispositif d'affichage comprend l'affichage dudit premier élément d'interface graphique utilisateur de façon qu'aucune partie de ce dernier ne soit cachée par ledit second élément d'interface graphique utilisateur qui est associé à un troisième processus exécuté dans ledit environnement d'exécution sécurisé.

4. Procédé selon la revendication 1, comprenant également :

l'affichage uniquement dudit premier élément d'interface graphique utilisateur sur ledit dispositif d'affichage lors de la réception d'une indication d'affichage sécurisé d'utilisateur.

5. Procédé selon la revendication 1, comprenant également :

l'acceptation d'une indication d'affichage de secret provenant d'un utilisateur ; et
l'affichage dudit premier secret en réponse à l'acceptation de ladite indication d'affichage de secret.

6. Procédé selon la revendication 1, comprenant également :

l'acceptation d'au moins deux éléments de données graphiques respectivement associés à un processus exécuté dans ledit environnement d'exécution sécurisé, en vue de leur affichage sur ledit dispositif d'affichage ; et

l'affichage d'au moins deux éléments d'interface graphique utilisateur, chacun desdits éléments d'interface graphique utilisateur comprenant l'un desdits éléments de données graphiques et une décoration d'interface graphique utilisateur commune.

7. Procédé selon la revendication 6, dans lequel ladite décoration d'interface graphique utilisateur commune comprend une bordure colorée.

8. Procédé selon la revendication 6, dans lequel ladite décoration d'interface graphique utilisateur commune comprend une ou plusieurs images sélectionnées de manière aléatoire.

9. Procédé selon la revendication 6, comprenant également :

le changement de ladite décoration d'interface graphique utilisateur commune lorsqu'une période de temps définie s'est écoulée.

10. Procédé selon la revendication 6, comprenant également :

le changement de ladite décoration d'interface graphique utilisateur commune lorsqu'une indication de changement de décoration d'utilisateur est reçue.

11. Procédé selon la revendication 1, dans lequel ledit premier élément d'interface graphique utilisateur comprend une fenêtre, ledit premier élément d'interface graphique utilisateur étant associé à des informations de libellé public et à des informations de libellé privé, dans lequel lesdites informations de libellé privé sont utilisées pour identifier ladite fenêtre dans ledit environnement d'exécution sécurisé, dans lequel lesdites informations de libellé privé ne sont pas connues dudit second environnement d'exécution, et dans lequel le procédé comprend également :

la fourniture desdites informations de libellé public pour qu'elles soient utilisées par ledit second environnement d'exécution afin d'identifier ladite fenêtre.

12. Procédé selon la revendication 11, dans lequel ledit second environnement d'exécution comprend un gestionnaire de fenêtre hôte pour gérer des éléments d'interface graphique utilisateur sur ledit dispositif d'affichage, dans lequel ledit gestionnaire de fenêtre hôte crée un élément d'interface graphique utilisateur avec un effet d'ombre pour ledit premier élément d'interface graphique utilisateur, et dans lequel ledit libellé public est utilisé par ledit gestionnai-

re de fenêtre hôte.

13. Support lisible par ordinateur contenant des instructions exécutables par ordinateur pour maintenir la sécurité de données affichées sur un dispositif d'affichage pour un système comportant un environnement d'exécution sécurisé et un second environnement d'exécution, les instructions exécutables par ordinateur étant adaptées pour réaliser des opérations comprenant :

le stockage d'une image d'au moins un premier élément d'interface graphique utilisateur associé à un premier processus exécuté dans ledit environnement d'exécution sécurisé ; et
l'affichage dudit premier élément d'interface graphique utilisateur qui comprend un premier secret associé audit environnement d'exécution sécurisé, sur ledit dispositif d'affichage entièrement sur un écran, de façon qu'aucune partie dudit premier élément d'interface graphique utilisateur ne soit cachée par un second élément d'interface graphique utilisateur associé audit second environnement d'exécution sur ledit dispositif d'affichage, ledit premier secret étant communiqué audit environnement d'exécution sécurisé par un utilisateur et n'étant pas accessible par un second processus exécuté dans ledit second environnement d'exécution.

14. Support lisible par ordinateur selon la revendication 13, dans lequel ladite opération d'affichage dudit premier élément d'interface graphique utilisateur comprend :

le fait de faire en sorte que ledit premier élément d'interface graphique utilisateur ne contienne aucune zone de transparence.

15. Support lisible par ordinateur selon la revendication 13, dans lequel ladite opération d'affichage dudit premier élément d'interface graphique utilisateur sur un dispositif d'affichage comprend l'affichage dudit premier élément d'interface graphique utilisateur de façon qu'aucune partie de ce dernier ne soit cachée par ledit second élément d'interface graphique utilisateur qui est associé à un troisième processus exécuté dans ledit environnement d'exécution sécurisé.

16. Support lisible par ordinateur selon la revendication 13, dans lequel les instructions exécutables par ordinateur sont adaptées pour réaliser des opérations comprenant également :

l'affichage uniquement dudit premier élément d'interface graphique utilisateur sur ledit dispositif d'affichage lors de la réception d'une indication d'affichage sécurisé d'utilisateur.

17. Support lisible par ordinateur selon la revendication 13, comprenant également :

l'acceptation d'une indication d'affichage de secret provenant d'un utilisateur ; et
l'affichage dudit premier secret en réponse à l'acceptation dudit affichage de secret.

18. Support lisible par ordinateur selon la revendication 13, dans lequel les instructions exécutables par ordinateur sont adaptées pour réaliser des opérations comprenant également :

l'acceptation d'au moins deux éléments de données graphiques respectivement associés à un processus exécuté dans ledit environnement d'exécution sécurisé, en vue de leur affichage sur ledit dispositif d'affichage ; et
l'affichage d'au moins deux éléments d'interface graphique utilisateur, chacun desdits éléments d'interface graphique utilisateur comprenant l'un desdits éléments de données graphiques et une décoration d'interface graphique utilisateur commune.

19. Support lisible par ordinateur selon la revendication 18, dans lequel ladite décoration d'interface graphique utilisateur commune comprend une bordure colorée.

20. Support lisible par ordinateur selon la revendication 18, dans lequel ladite décoration d'interface graphique utilisateur commune comprend une ou plusieurs images sélectionnées de manière aléatoire.

21. Support lisible par ordinateur selon la revendication 18, dans lequel les instructions exécutables par ordinateur sont adaptées pour réaliser des opérations comprenant également :

le changement de ladite décoration d'interface graphique utilisateur commune lorsqu'une période de temps définie s'est écoulée.

22. Support lisible par ordinateur selon la revendication 18, dans lequel les instructions exécutables par ordinateur sont adaptées pour réaliser des opérations comprenant également :

le changement de ladite décoration d'interface graphique utilisateur commune lorsqu'une indication de changement de décoration d'utilisateur est reçue.

23. Support lisible par ordinateur selon la revendication 13, dans lequel ledit premier élément d'interface graphique utilisateur comprend une fenêtre, ledit premier élément d'interface graphique utilisateur étant

associé à des informations de libellé public et à des informations de libellé privé, dans lequel lesdites informations de libellé privé sont utilisées pour identifier ladite fenêtre dans ledit environnement d'exécution sécurisé, dans lequel lesdites informations de libellé privé ne sont pas connues dudit second environnement d'exécution, et dans lequel les opérations réalisables par les instructions exécutables par ordinateur comprennent également :

la fourniture desdites informations de libellé public pour qu'elles soient utilisées par ledit second environnement d'exécution afin d'identifier ladite fenêtre.

24. Support lisible par ordinateur selon la revendication 23, dans lequel ledit second environnement d'exécution comprend un gestionnaire de fenêtre hôte pour gérer des éléments d'interface graphique utilisateur sur ledit dispositif d'affichage, dans lequel ledit gestionnaire de fenêtre hôte crée un élément d'interface graphique utilisateur avec un effet d'ombre pour ledit premier élément d'interface graphique utilisateur, et dans lequel ledit libellé public est utilisé par ledit gestionnaire de fenêtre hôte.

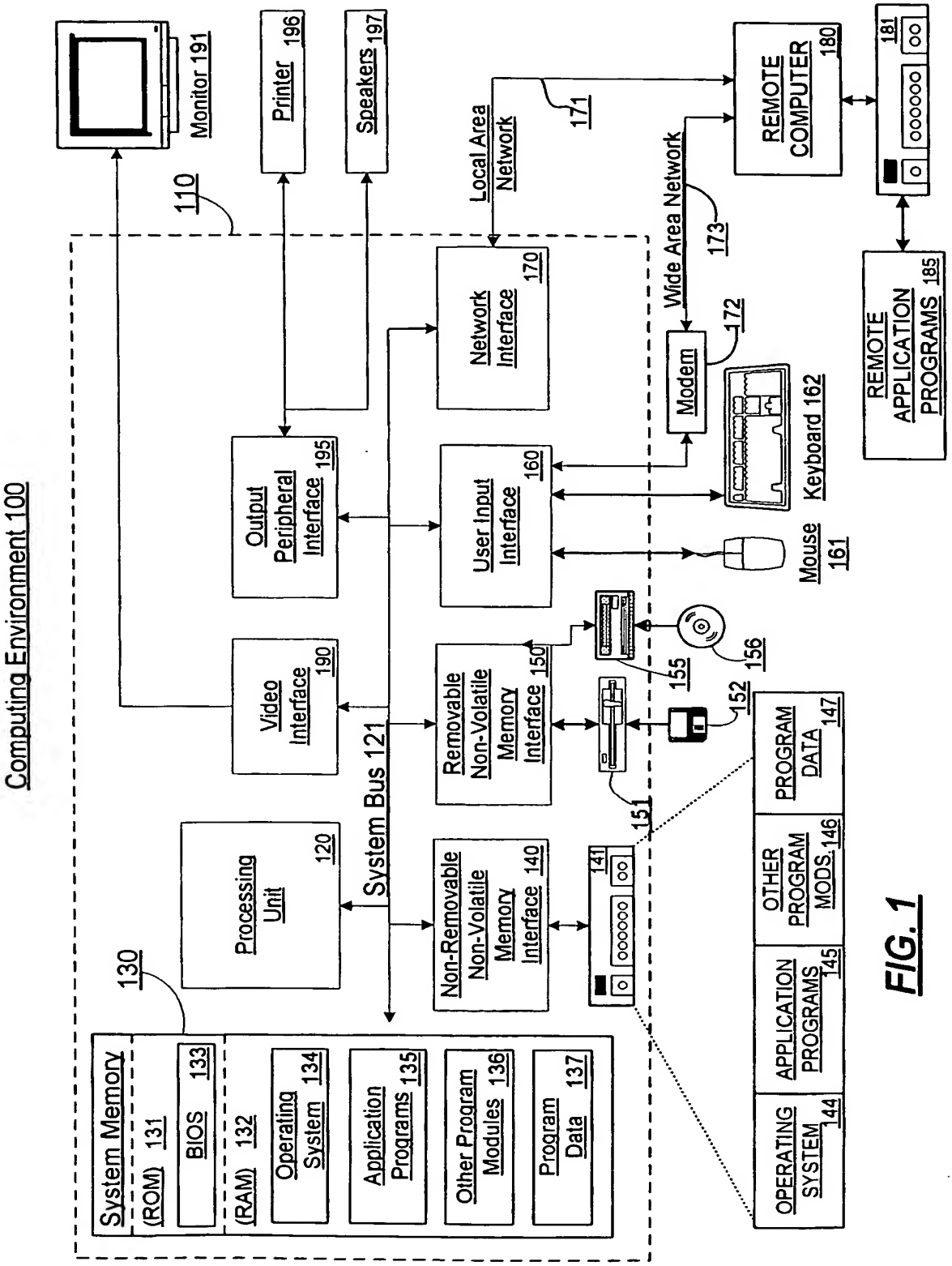
Computing Environment 100
110
Monitor 191
Printer 196
Speakers 197
Local Area Network
171
REMOTE COMPUTER 180
181
Wide Area Network
173
Network Interface 170
Modem 172
REMOTE APPLICATION PROGRAMS 185
Output Peripheral Interface 195
User Input Interface 160
Keyboard 162
Mouse 161
Video Interface 190
Removable Non-Volatile Memory Interface 150
155
156
152
151
System Bus 121
Processing Unit 120
Non-Removable Non-Volatile Memory Interface 140
141
130
System Memory
(ROM) 131
BIOS 133
(RAM) 132
Operating System 134
Application Programs 135
Other Program Modules 136
Program Data 137
OPERATING SYSTEM 144
APPLICATION PROGRAMS 145
OTHER PROGRAM MODS. 146
PROGRAM DATA 147

FIG. 1

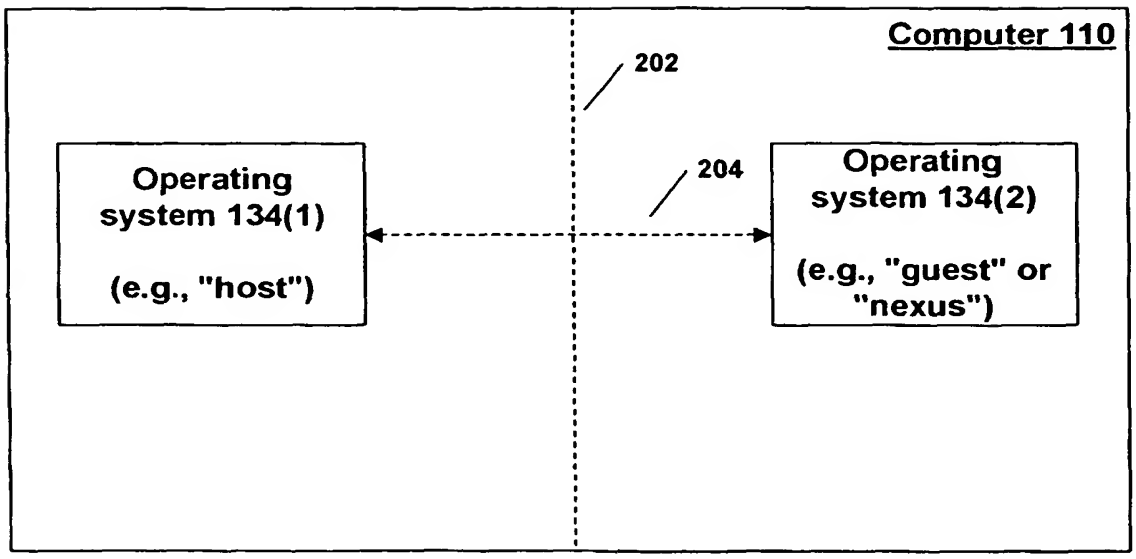
Computer 110
202
204
Operating system 134(1)
(e.g., "host")
Operating system 134(2)
(e.g., "guest" or "nexus")

FIG. 2

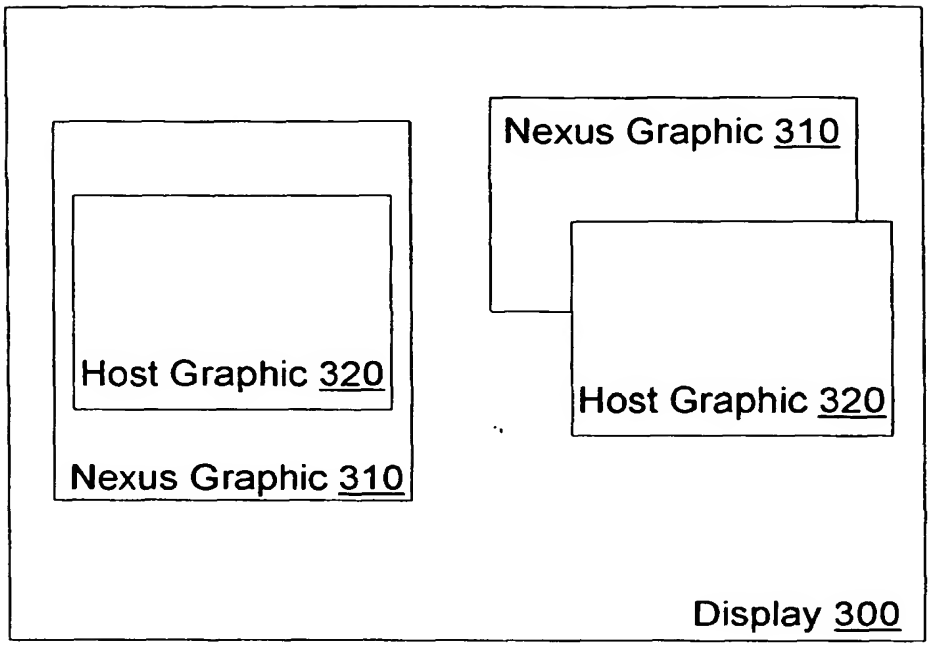
Nexus Graphic 310
Host Graphic 320
Host Graphic 320
Nexus Graphic 310
Display 300

FIG. 3(a)

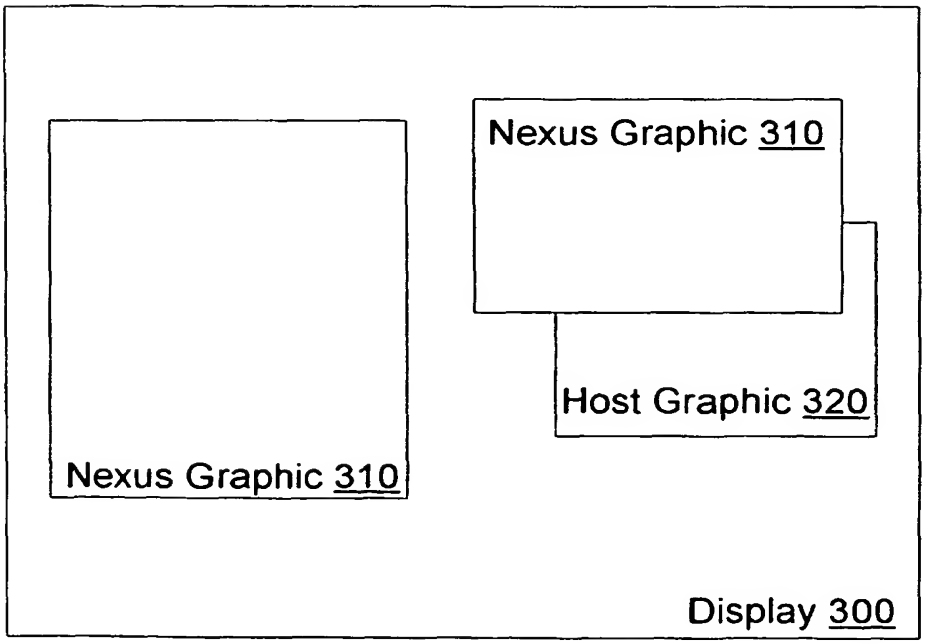
Nexus Graphic 310
Host Graphic 320
Nexus Graphic 310
Display 300

FIG. 3(b)

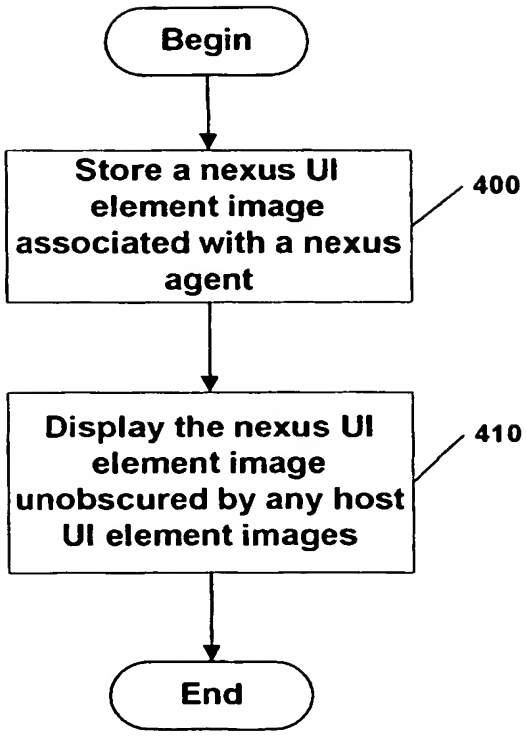
Begin
Store a nexus UI element image associated with a nexus agent
400
Display the nexus UI element image unobscured by any host UI element images
410
End

FIG. 4

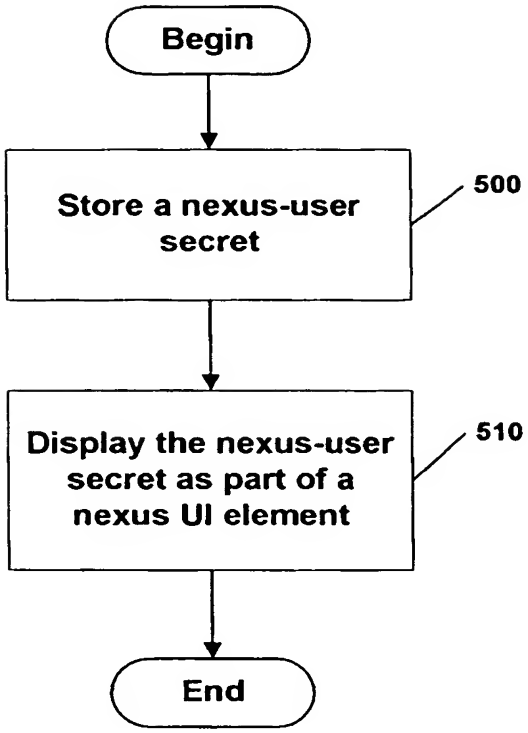
Begin
Store a nexus-user secret
500
Display the nexus-user secret as part of a nexus UI element
510
End

FIG. 5

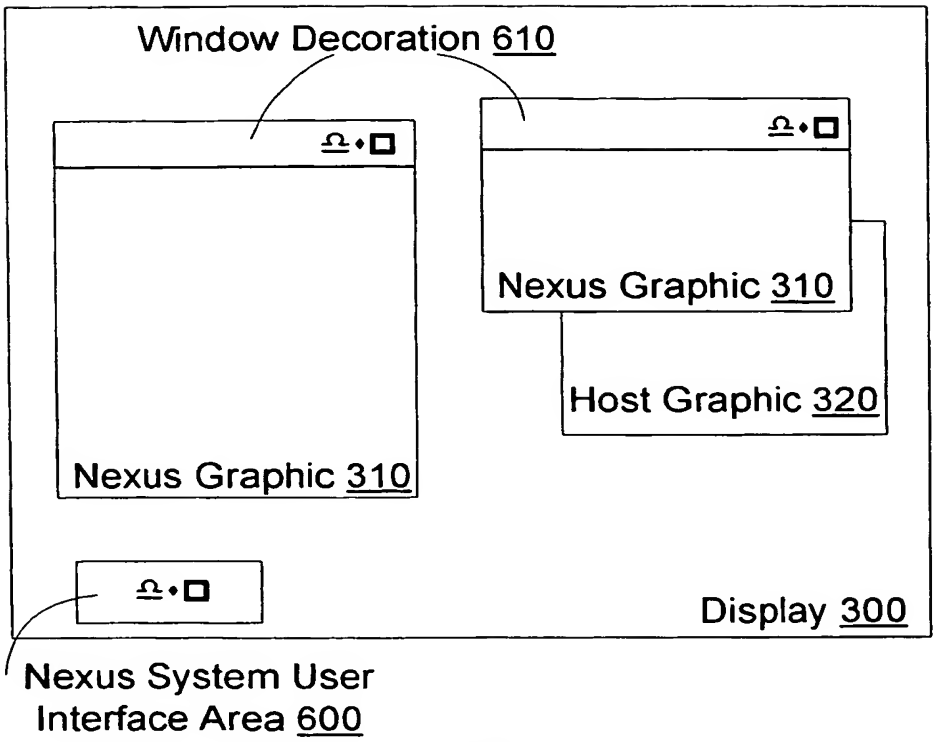
Window Decoration 610
Nexus Graphic 310
Host Graphic 320
Nexus Graphic 310
Display 300
Nexus System User Interface Area 600

FIG. 6

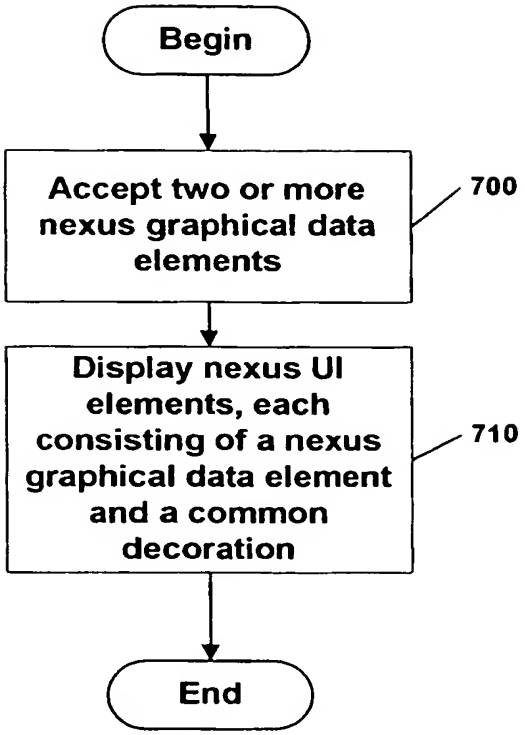
Begin
Accept two or more nexus graphical data elements
700
Display nexus UI elements, each consisting of a nexus graphical data element and a common decoration
710
End

FIG. 7

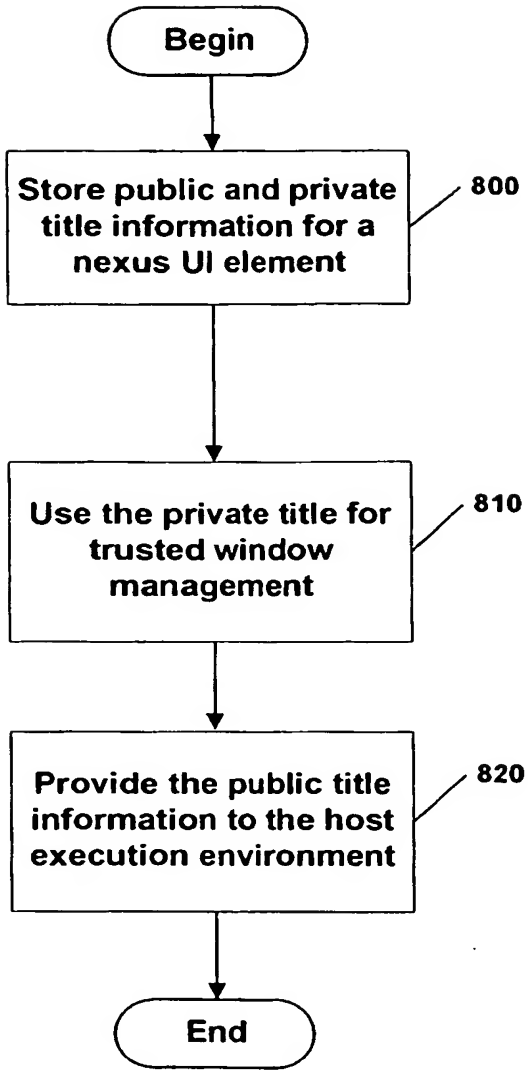
Begin
Store public and private title information for a nexus UI element
800
Use the private title for trusted window management
810
Provide the public title information to the host execution environment
820
End

FIG. 8